IUT de Paris Descartes – Base de la Programmation Objet

Travaux Pratiques – Sujet n°2

Objectifs

- Savoir faire suivre, attraper et traiter des exceptions
- Maîtriser la lecture de fichier texte
- Savoir manipuler des chaînes de caractères

Travail à faire

Vous disposez du type suivant (le fichier correspondant est sur le serveur) :

```
package artistes;

public class Personne {
  String prénom, nom;
}
```

Notez qu'il est défini dans le paquetage nommé `artistes`. Vous disposez aussi d'un fichier texte nommé `artistes.txt` contenant les prénoms et noms de personnes connues.

1. Créez un nouveau projet, importez-y le type `Personne` (au sein du paquetage `artistes`) ainsi que le fichier texte. Ce dernier doit être déposé à la racine du projet.
2. Dans une nouvelle classe (nommée `Artiste`), écrivez une fonction permettant de normaliser le prénom et le nom d'une personne. Pour ce faire, vous éliminerez les espaces en début et en fin de chaîne. De plus, la première lettre doit être en majuscule et les suivantes en minuscule. Notez bien que votre fonction doit supporter les chaînes vides.

   Pour réaliser ce traitement, vous pouvez vous baser sur la documentation des classes `String` et `Character` et, en particulier, les méthodes `trim`, `isEmpty`, `charAt`, `substring`, `toLowerCase` et `toUpperCase` de la classe `String`.
3. Écrivez un court programme déclarant une personne, initialisant son prénom et son nom et les affichant avant et après normalisation. Vous pouvez incorporer des espaces, des tabulations ('\t') et des retours à la ligne ('\n') en début et en fin de chaîne ainsi qu'alterner majuscules et minuscules pour observer la qualité de votre fonction de normalisation. Vérifier aussi que les chaînes vides ou composées d'un unique caractère sont supportées.
4. Le fichier `artistes.txt` contient une liste d'artistes. Chaque ligne du fichier précise le prénom et le nom d'un artiste. Développez une fonction lisant les personnes une à une et retournant la liste de personnes correspondantes. La liste retournée doit comporter uniquement des personnes dont le prénom et le nom ont été normalisés. Votre fonction doit être paramétrée par le nom du fichier devant être lu et lever une exception de type `FileNotFoundException` si le fichier n'existe pas.

   Il vous est recommandé de lire le fichier ligne à ligne (en vous inspirant du transparent 45 du support de cours) et d'analyser chaque ligne lue pour en extraire le prénom et le nom de chaque personne. Notez

que la classe `Scanner` (et ses méthodes `hasNext()` et `next()`) peut être employée pour lire mot à mot le contenu d'une chaîne de caractère. Un scanner `sc` permettant d'analyser une chaîne `s` est obtenu par `sc = new Scanner(s)`.

5. Complétez le programme de manière à lire le fichier `artistes.txt` et afficher le nombre de personnes lues. Si le fichier ne peut être ouvert, votre programme doit afficher le message "*le fichier est introuvable*" et se terminer proprement.

Étude de performances – questions optionnelles

6. Écrivez une fonction retournant le temps nécessaire à la constitution d'une chaîne de caractère comportant les noms et prénoms de 10000 personnes (une par ligne). Les personnes seront prises au sein d'une liste (non-vide) de personnes. Si la liste ne contient pas suffisamment de personnes, des noms issus de cette liste seront répétés autant de fois que nécessaire. La chaîne de caractères doit être constituée par des concaténations successives (opérateur +). Complétez le programme de façon à afficher le temps de calcul de votre fonction. Les instructions suivantes montrent comment le temps de calcul peut être déterminé :

```
long début = System.currentTimeMillis();
// calcul ...
long fin = System.currentTimeMillis();
long durée = fin - début; // en millisecondes
```

7. L'opérateur de concaténation n'est pas une bonne solution lorsque le nombre de chaînes de caractère intermédiaires devient important. Le type `StringBuilder` permet de modifier des chaînes existantes alors que le type `String` ne l'autorise pas. Les instructions suivantes illustrent les éléments nécessaires à l'optimisation de la fonction précédente :

```
StringBuffer sb = new StringBuffer(); // une chaîne modifiable vide
sb.append("opti"); // concaténation
sb.append("misons"); // concaténation
String s = sb.toString(); // conversion en une chaîne non-modifiable
```

Corrigez la fonction précédente en conséquence.